Process - Oriented Approach And Data Flow Diagrams

Structuring Systems Requirements

Process-Oriented Approach

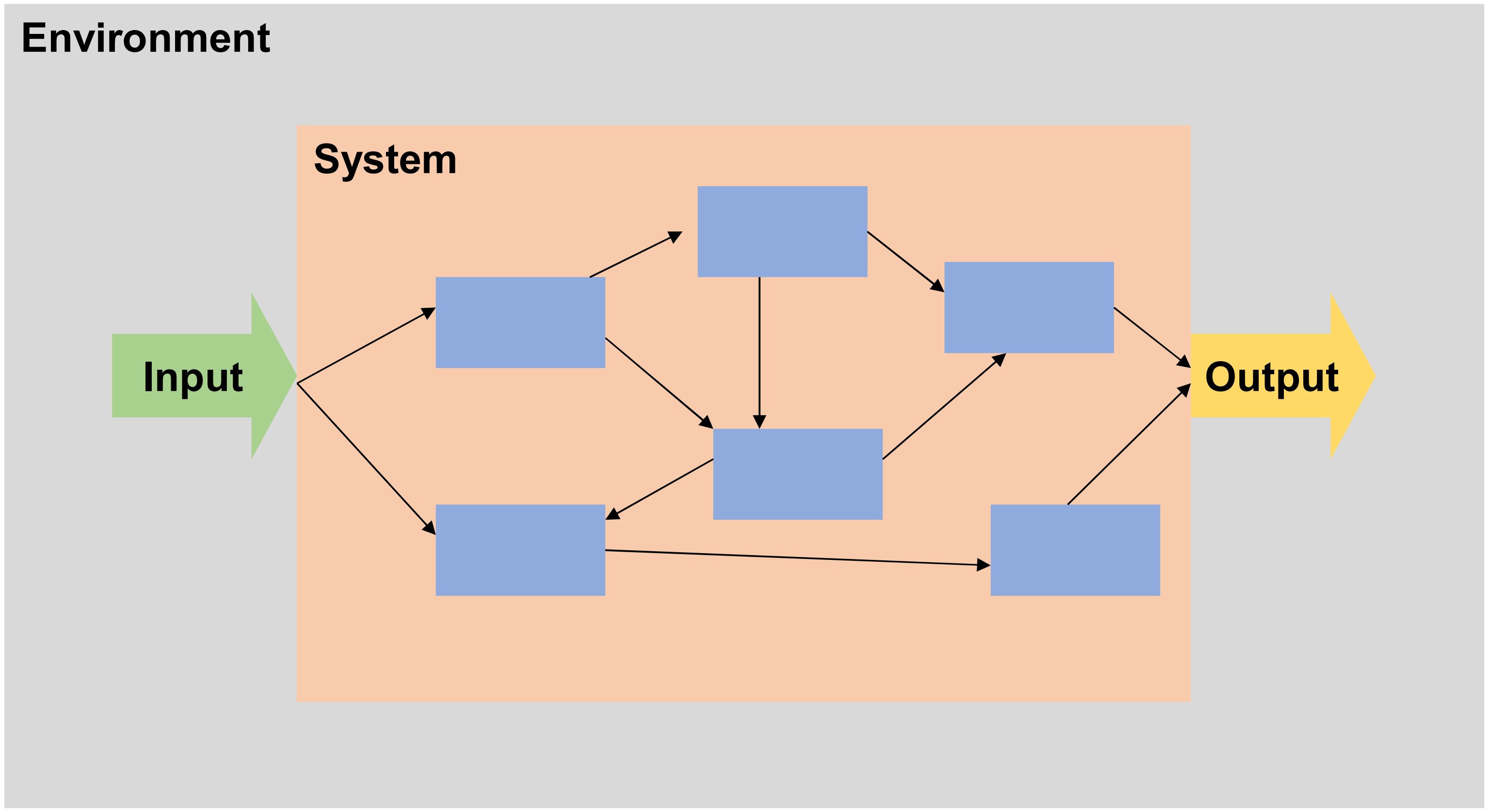
- **Process Modeling**
- **Data Modeling**
- **Object Modeling**

Data-Oriented Approach

Object-Oriented Approach

Process-Oriented Approach

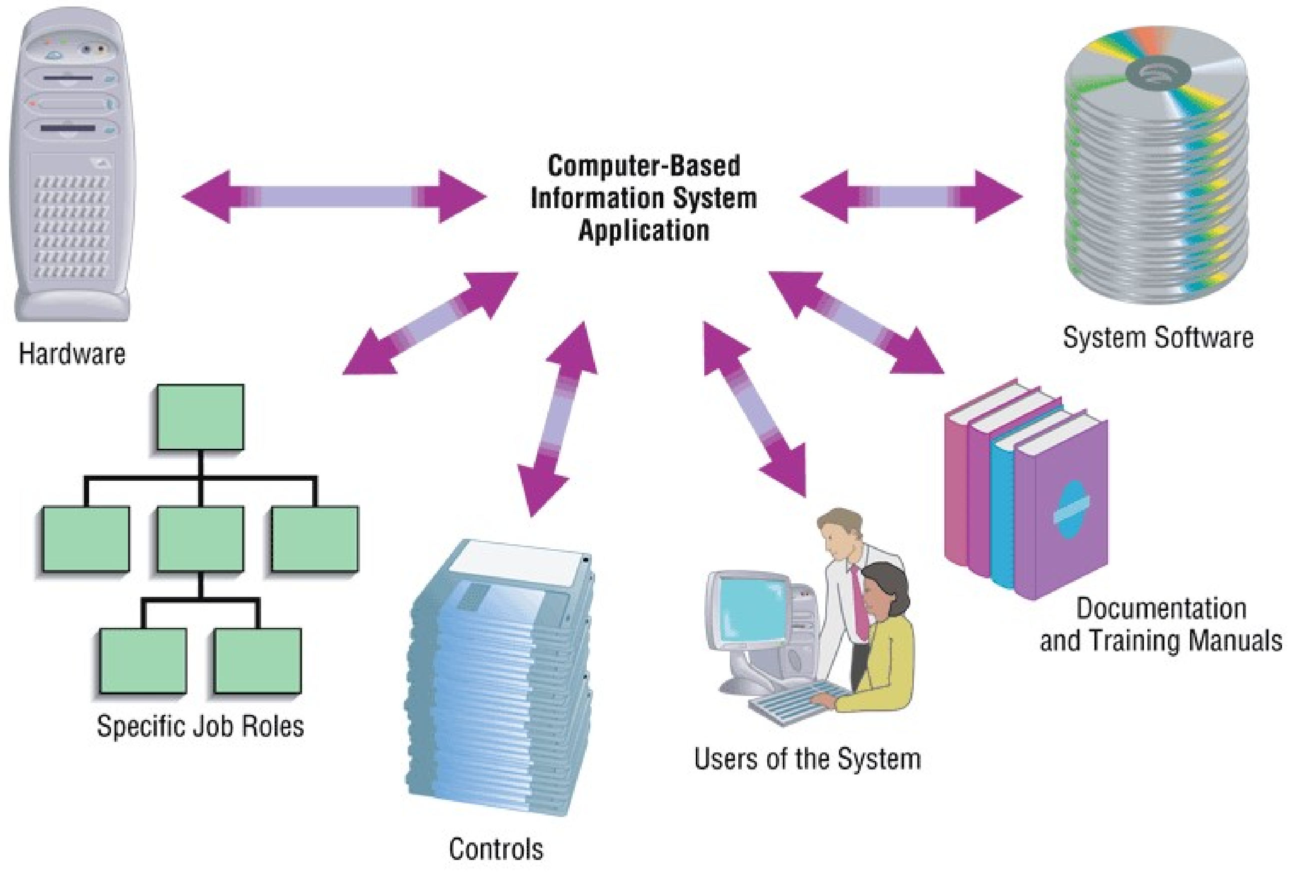
A business entity can be thought of as a system.



System

- System: Turns data into information and includes:
  - Hardware and system software
  - Documentation and training materials
  - Job roles associated with the system
  - Controls to prevent theft or fraud
  - The people who use the software to perform their jobs

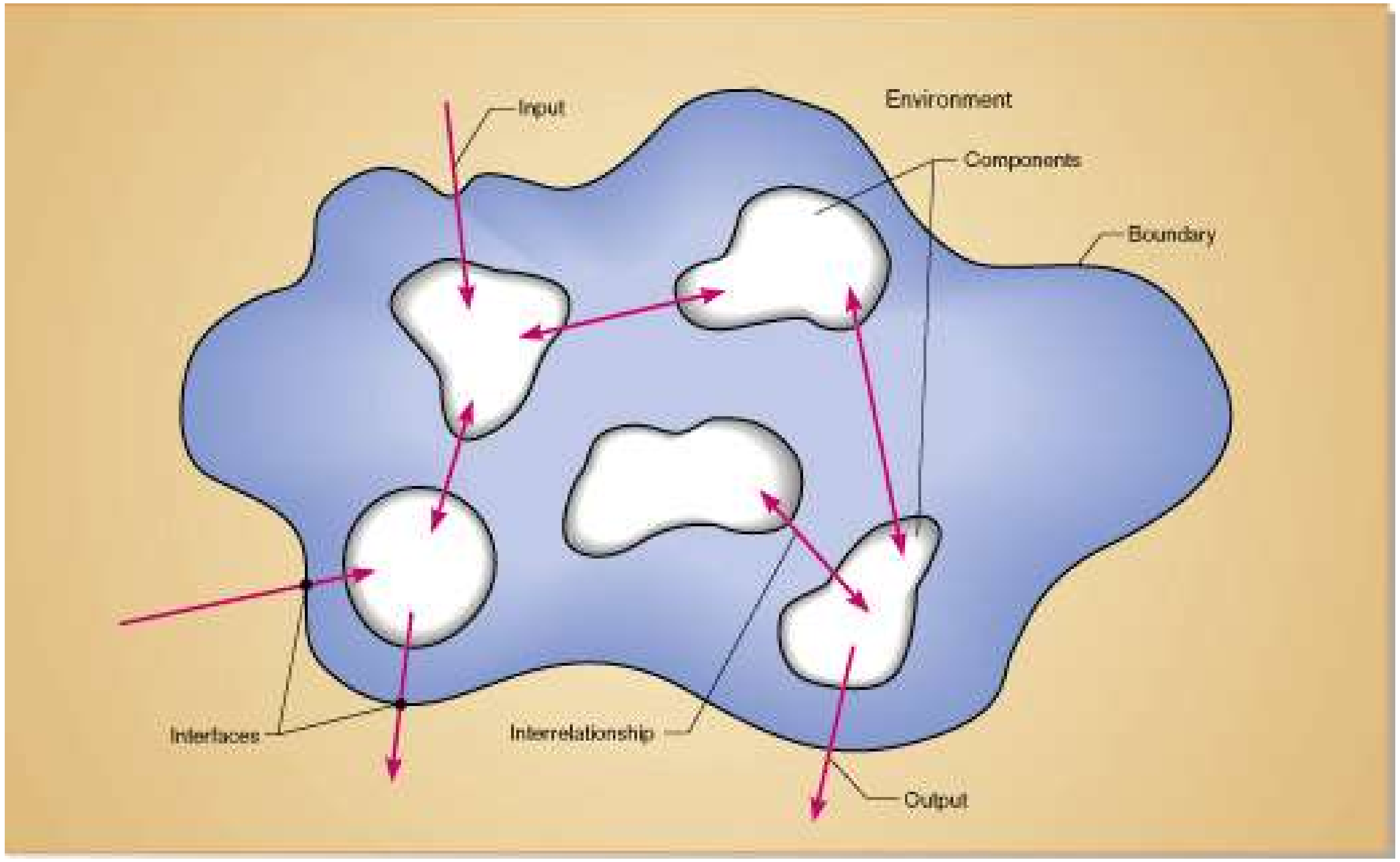
System


Computer-Based Information System Application
Hardware
System Software
Specific Job Roles
Controls
Users of the System
Documentation and Training Manuals

System

- A system is an interrelated set of business procedures used within one business unit working together for a purpose
- A system has nine characteristics
- A system exists within an environment
- A boundary separates a system from its environment

Characteristics of a System



Important System Concepts

- Decomposition
  - The process of breaking down a system into smaller components
  - Allows the systems analyst to:
    - Break a system into small, manageable subsystems
    - Focus on one area at a time
    - Concentrate on component pertinent to one group of users
    - Build different components at independent times

Important System Concepts

- Modularity
  - Process of dividing a system into modules of a relatively uniform size
  - Modules simplify system design
- Coupling
  - Subsystems that are dependent upon each other are coupled

Process-Oriented Approach

- Three key components of an information system
- **Data vs. Information**
    - **Data**
        - Raw facts
    - **Information**
        - Information is data that has been organized and interpreted, and possibly formatted, filtered, analyzed, and summarized
        - Derived from data
        - Organized in a manner that humans can understand

Process-Oriented Approach

- Data
  - Understanding the source and use of data is key to good system design
  - Various techniques are used to describe data and the relationship amongst data
- Data Flows
  - Groups of data that move and flow through the system

Process-Oriented Approach

- Data Flows (Continued)
  - Include description of sources and destination for each data flow
- Processing Logic
  - Describe steps that transform data and events that trigger the steps

Process-Oriented Approach

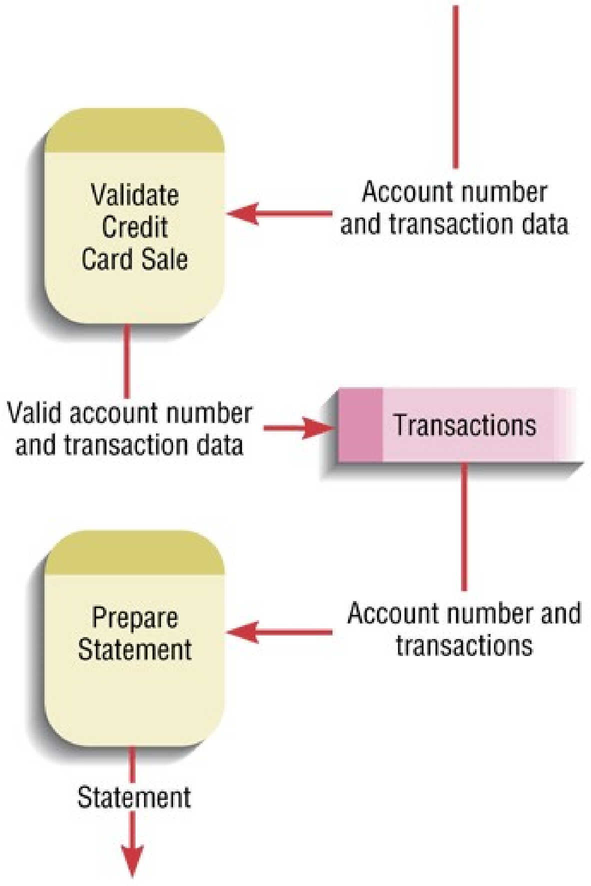
Figure 1.8 Differences among Data, Data Flow, and Processing Logic

Data

Name	Age	Party
John Smith	25	Democrat
Joan Chen	42	Republican
Wilma Alvarez	31	Independent

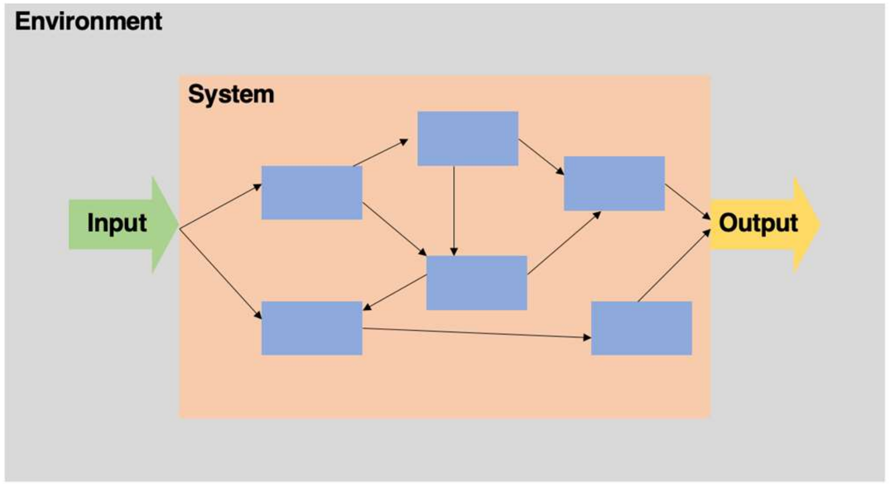
Processing Logic

```
Event::Hours-Worked = 0
Event Action::
If Hours-Worked > 40 then
    Pay = 40 * Pay-Rate
    + (Hours-Worked - 40) * (1.5 * Pay-Rate)
Else
    Pay = Pay-Rate * Hours-Worked
End if
```


Validate Credit Card Sale
Account number and transaction data
Valid account number and transaction data
Transactions
Prepare Statement
Account number and transactions
Statement

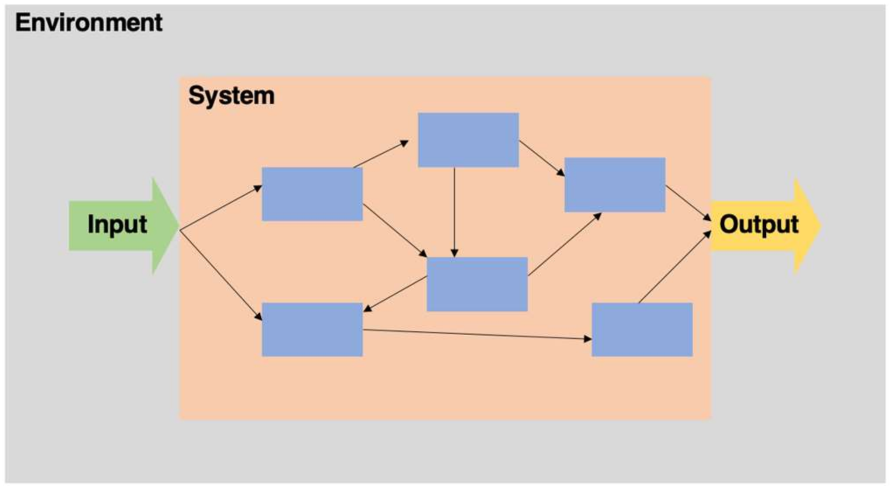
Process-Oriented Approach

- An Organization performs several processes to accomplish its objectives
- These processes can be quite complex for human beings to comprehend
- There is therefore a need to model these processes



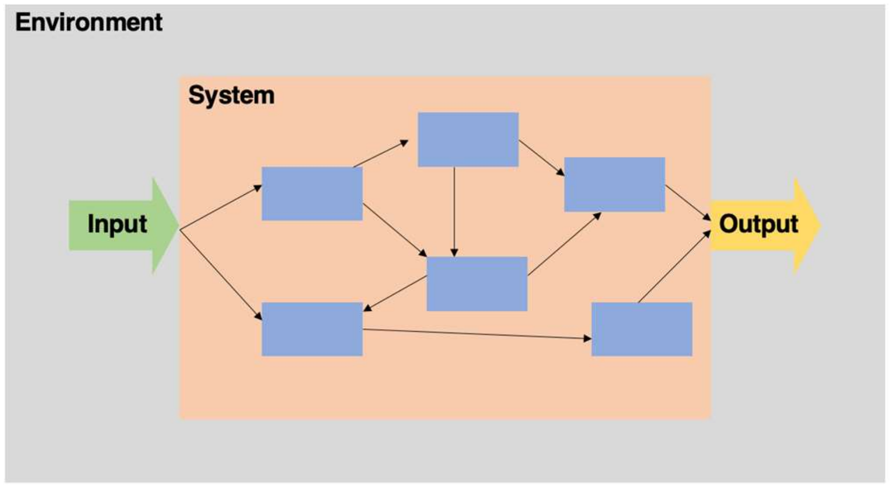
Process-Oriented Approach

- At a given point in time, a system has a state.
- A system goes from state to state.
- What causes the state of a system to change?
- A process
    - A transaction
    - An authorization
    - A data entry
    - A purchase
    - A shipment
    - A manufacturing task



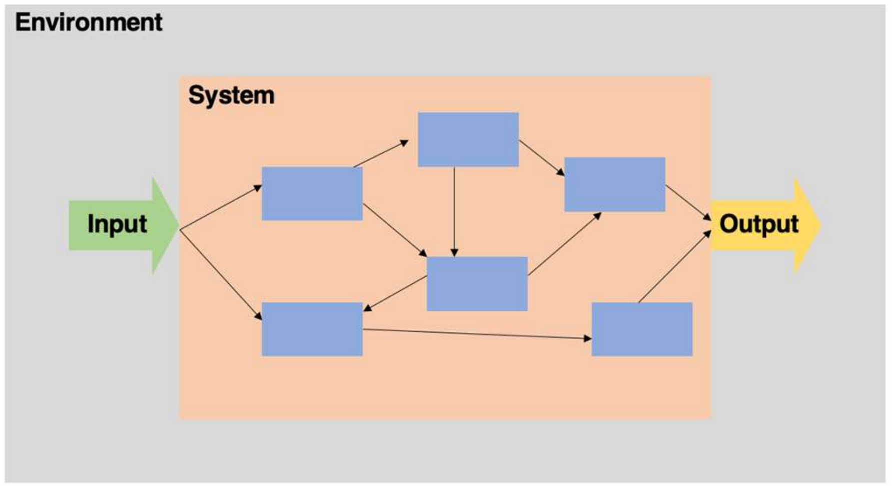
Process-Oriented Approach

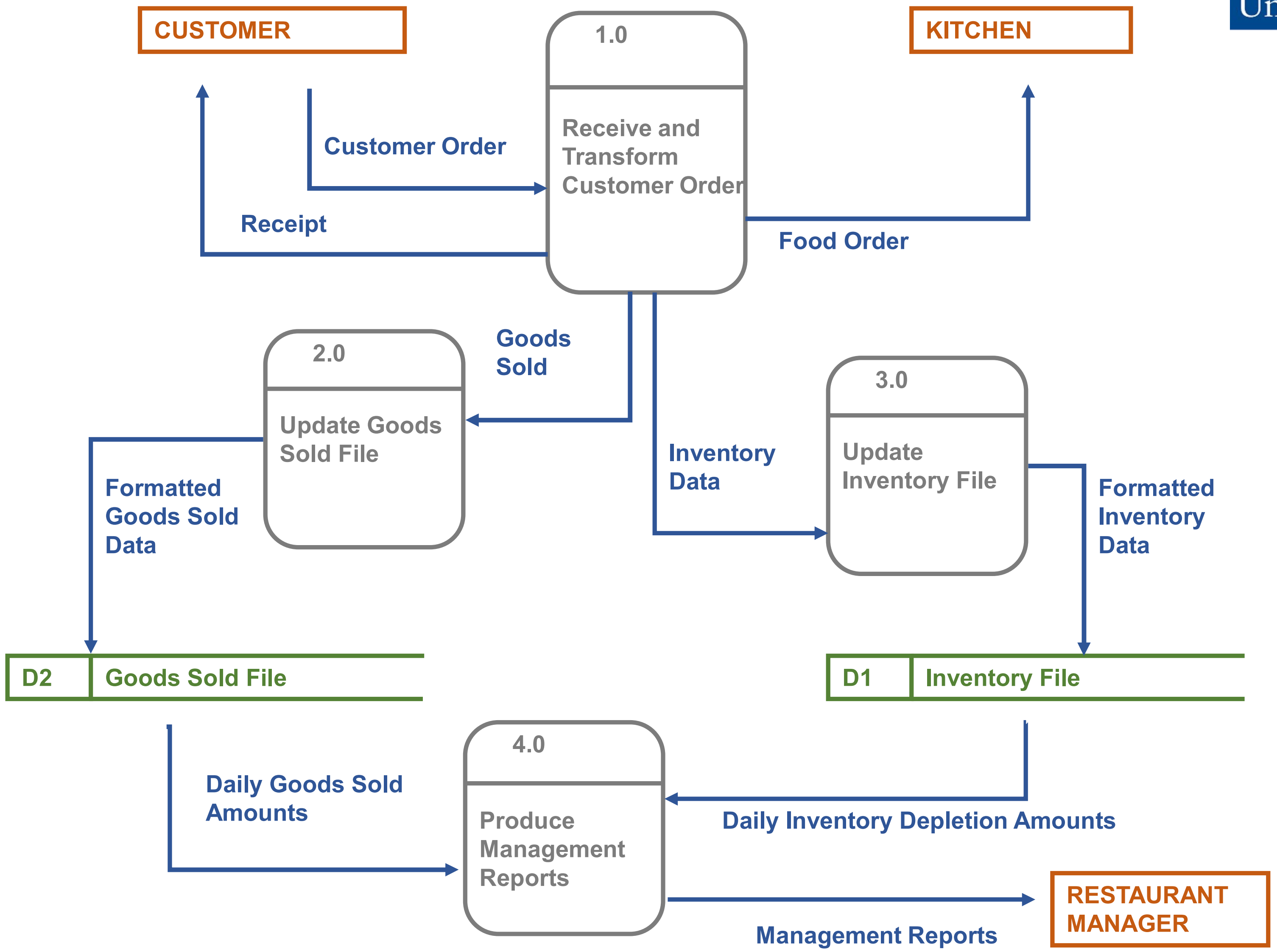
- Identifying and graphically modeling each process is Process Modeling.
- A process can be thought of as a system and vice versa.
- So a process has an input and an output.
- The input could be a data item or a tangible thing like a piece of paper or machinery.



Data Flow Diagrams

- We can represent physical or non-physical entities in the system with data.
- As the data changes using some process, the state of the system changes.
- The change and flow of data through processes can be modeled using data flow diagrams.




CUSTOMER
KITCHEN
1.0
Receive and Transform Customer Order
Customer Order
Receipt
Food Order
Goods Sold
2.0
Update Goods Sold File
Inventory Data
3.0
Update Inventory File
Formatted Goods Sold Data
Formatted Inventory Data
D2
Goods Sold File
D1
Inventory File
4.0
Produce Management Reports
Daily Goods Sold Amounts
Daily Inventory Depletion Amounts
Management Reports
RESTAURANT MANAGER

Few Observations

Data is not flowing all the time

Which implies sometimes Data is at rest

- (Data Store)

Something should cause data to move

- (Process)

Data comes into the system from somewhere

- (Source)

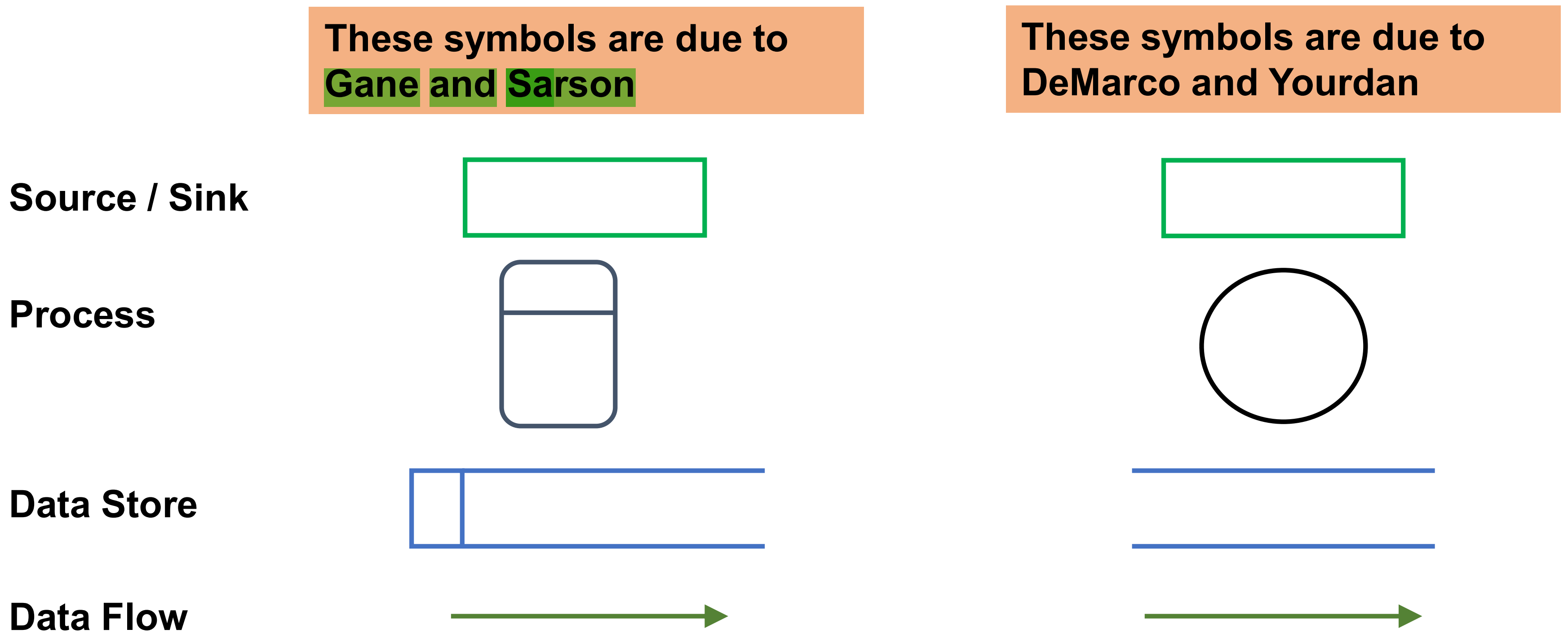
Data leaves the system and goes somewhere

- (Sink)

Data flows between Source/ Sink/ Data Store/ Process (Data Flows)

Data Flow Diagrams (DFDs)

Using Source/Sink, Process, Data Stores and Data Flows, we can show the flow of data coming into the system, within the system and going out of the system.


These symbols are due to Gane and Sarson
These symbols are due to DeMarco and Yourdan
Source / Sink
Process
Data Store
Data Flow

What is a source or a sink ?

- A Source provides the origin of data.
  - *e.g. Customer* may provide Order data
  - *e.g. Employee* may provide “Hours Worked” data
- A Sink provides destination of data
  - *e.g.* Reports go to *Manager*
  - *e.g.* Order data goes to *Shipping dept*
  - *e.g.* Paycheck goes to *Employee*
- Sources and Sink are considered outside the system of interest. They define the boundary of the system.
- Sources and sinks do <u>not</u> have direct access to data stores!!!!
- They have to be identified nevertheless, to understand the context in which the system has to function.

What are Processes?

- A task or set of tasks performed on data by a person, a machine, or a computer so that the data are transformed, stored or distributed
- Input data flows are transformed by process and become output data flows

What do Processes do ?

Processes

- capture data from sources
  - e.g. the process of taking customer order
- produce/distribute data to sinks
  - e.g. the process that generates management reports
- maintain Data stores
  - e.g. the process of updating inventory data
- High-level descriptions of data transformation operations
  - e.g. the process of calculating invoice total

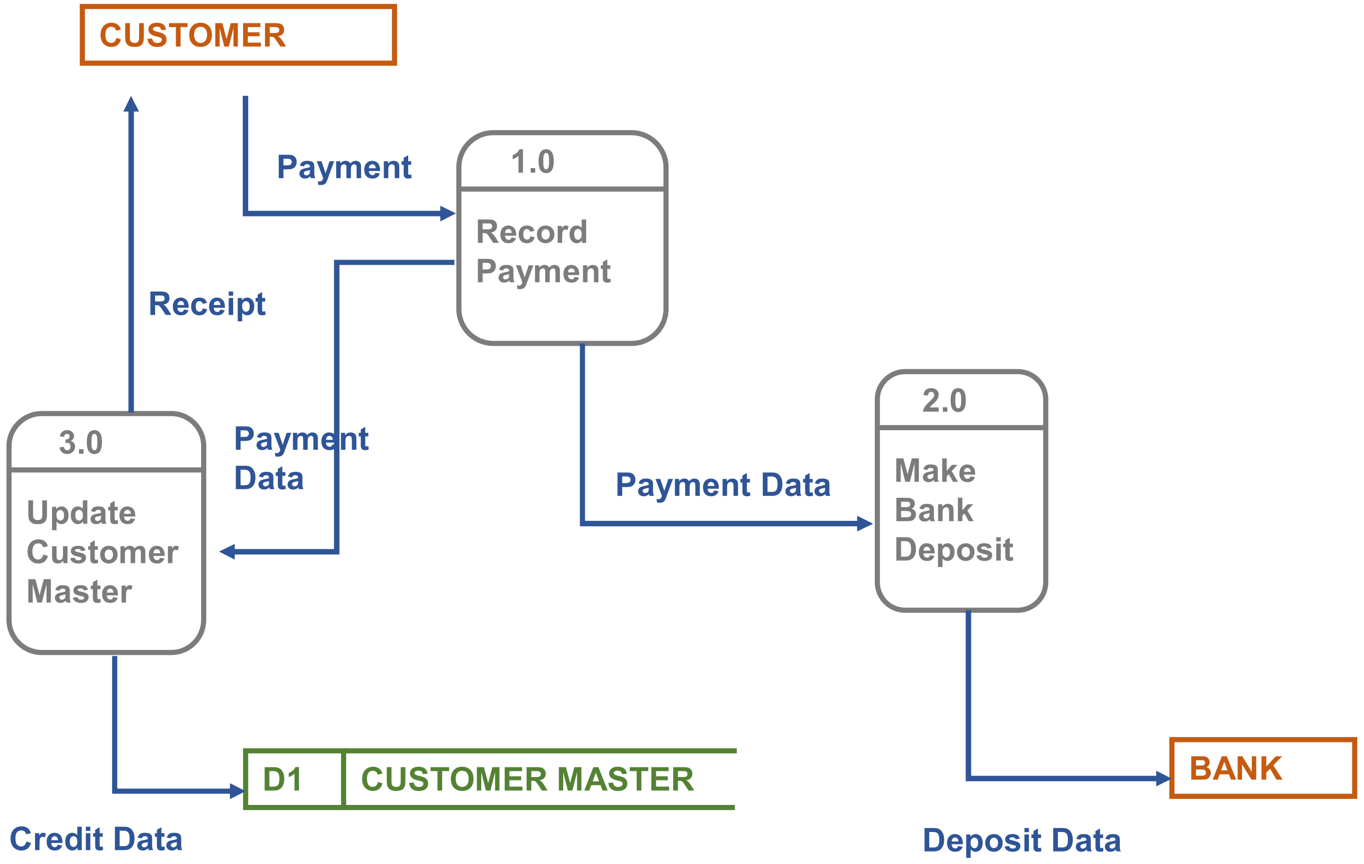
What are Data Stores?

- Repository for data temporarily or permanently recorded within the system
- Data at Rest is a Data Store
  - e.g. The Inventory Master File
  - e.g. The Customer Master File
  - e.g. The Sales Transaction File
  - e.g. The General Ledger File
- Data Stores may be on a variety of media such as Hard Disk, CD, Notebook, Manila Folder
- Direction of data flow ‘to’ or ‘from’ data store designates whether data is written into or read from data store

What are Data Flows?

- A data flow represents data being conveyed to or from an external entity, a process, or data store
- Data in motion (while going from one point in a system to another) is Data Flow
    - e.g. data flows from Payroll Transaction Files to a payroll check.
    - e.g. data flows from Customer’s mouth to a Order file
    - e.g. data flows from several data stores onto a report or a query

Another Sample DFD



Recap

- Data flows from a source into the system (input)
- It flows from the system out to a sink (output)
- While inside the system, data flows between data stores, and that
- Processes cause data to flow between source, data stores and sink.

How do we go about building a DFD?

Hierarchical Structure

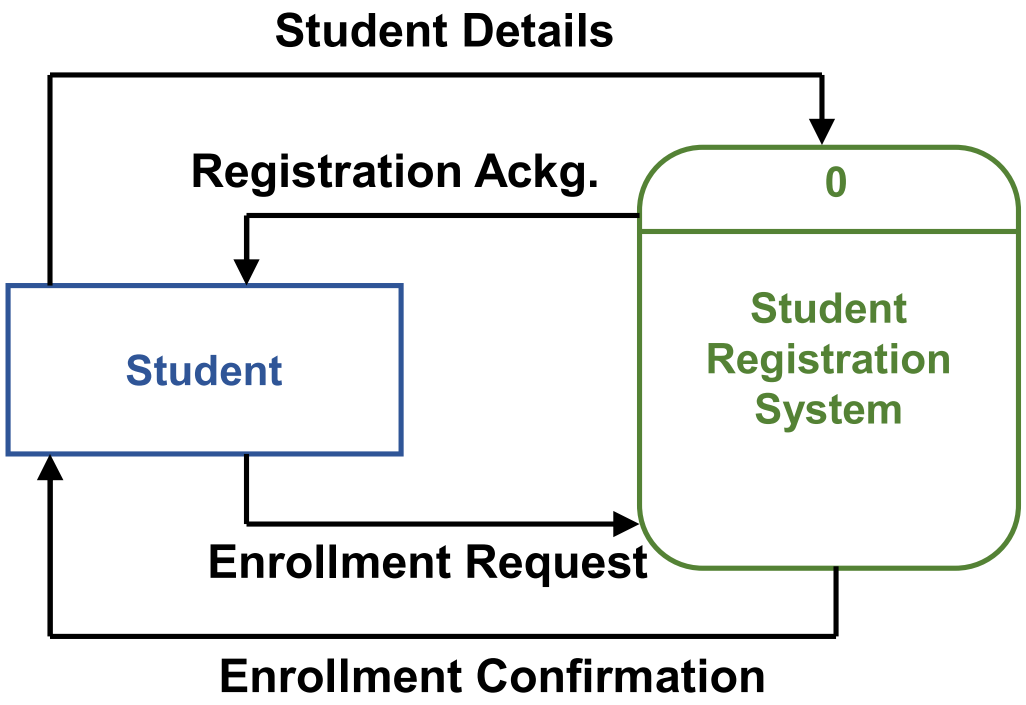
- Data flow diagram is organized in a hierarchical framework of levels
  - Each successive level reveals further details of the system
- Model of system begins with Context Level DFD
  - Level 0 diagram
    - Level 1 diagram
      - Level 2, 3 ….

Start with the Context Diagram

- Highest-level view of the system
- Contain only one process labeled “0”
- No data stores (contained within the process)
- A data flow diagram (DFD) of the scope of an organizational system that shows the system boundaries, external entities that interact with the system and the major information flows between the entities and the system

Start with the Context Diagram

- Here is an example of a context diagram



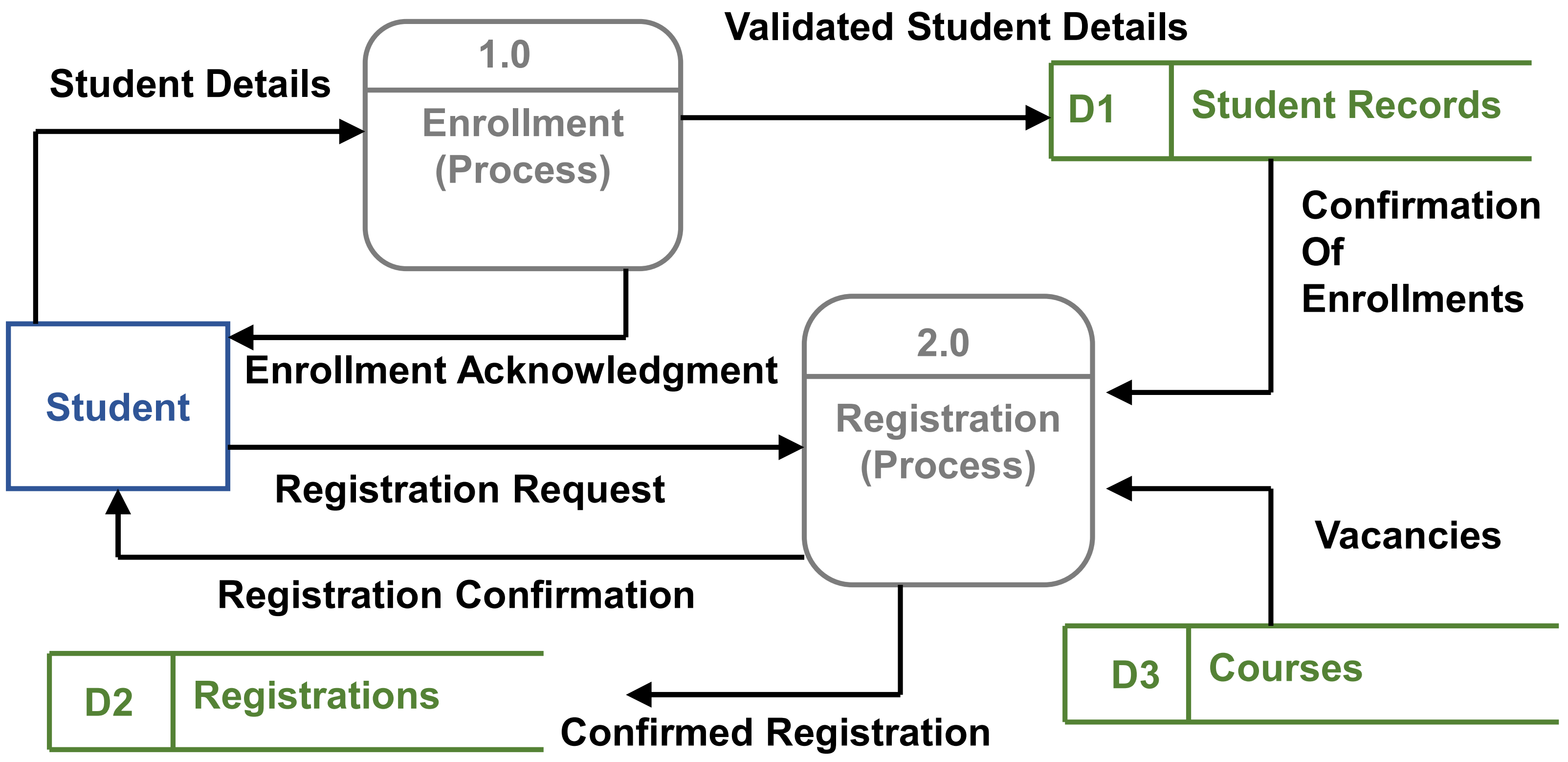
Contex Diagram :
1 tane sistem, entitiler ve input output yer almaldr. System 0 olarak adlandrlmaldr.// only one process
Level 0 Diagram :
Farkl processler data ve data storlar bulunabilir genel ak belirtilir
Level 1 Diagram :
- Detayl olarak bir ak belirtilir.

Level-0 Diagram

- Context level “process” is *exploded*
- Represents a system’s major processes, data flows, and data stores at a high level of detail
- Have same sources/sinks as context level diagram
- Each process number ends in “ .0”

Next, we Decompose

- You break up (or Decompose) the Context Diagram to what is called a **Level-0 Diagram**



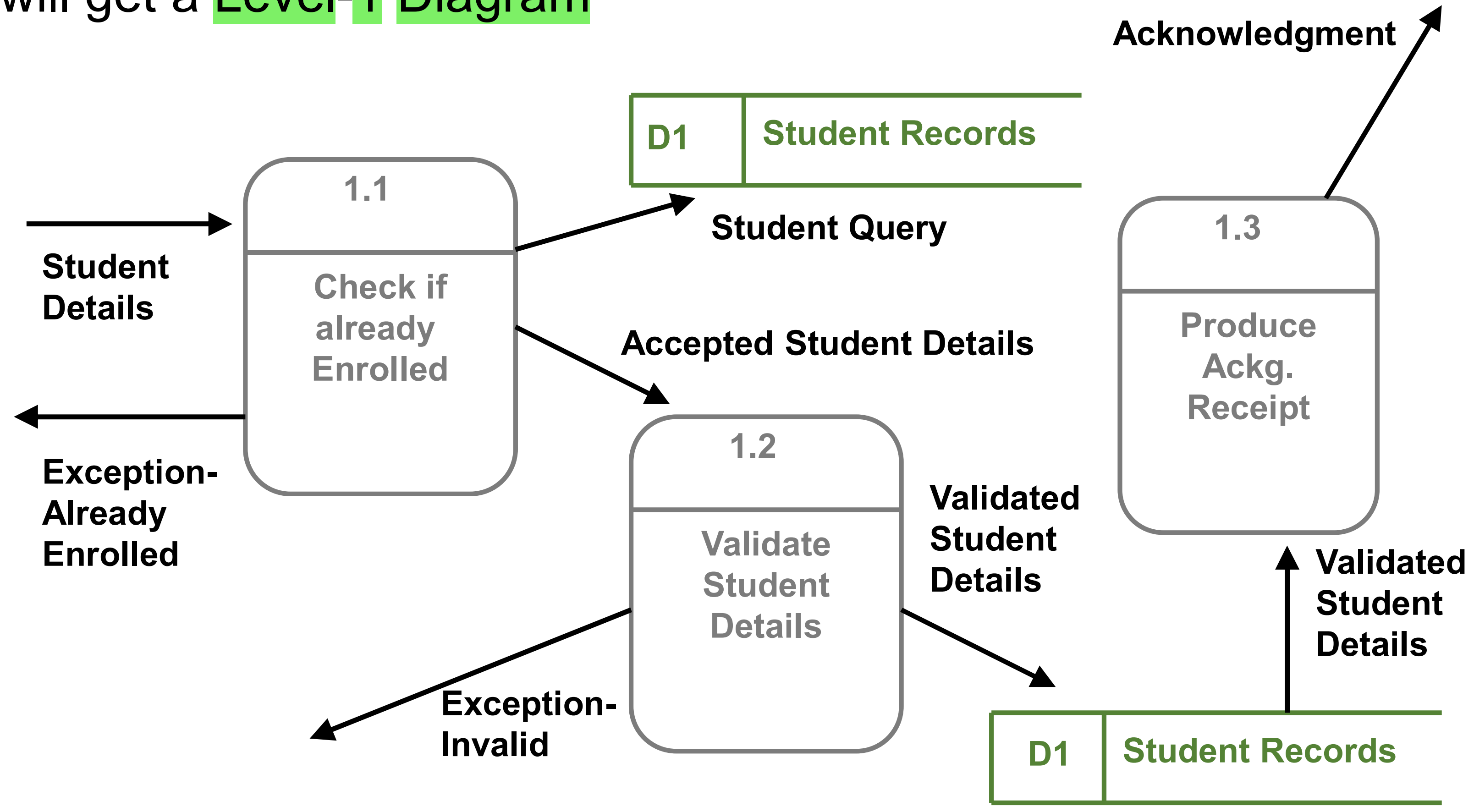
Decompose Further

- Each process can be decomposed further.
- Each process on level 0 diagram is exploded to deeper levels as necessary
  - inputs and outputs shown must be the same at successive DFD levels
- Consistency of Levels
  - The extent to which information contained on one level of a set of nested DFDs is also included on other levels.
- If we decompose a process in a Level-0 Diagram, we go to a Level-1 Diagram.
- Let’s look at an example.

Sabancı Universitesi SABANCI BUSINESS SCHOOL

Let us decompose the Enrollment Process further

We will get a Level-1 Diagram



This ensures that all the required data (e.g., name, ID, qualifications) is accurate and complete.

Decompose Further?

- Sure
- You will get a level-2 Diagram
- You decompose process 1.1 into three parts

How far can we go with this decomposition?

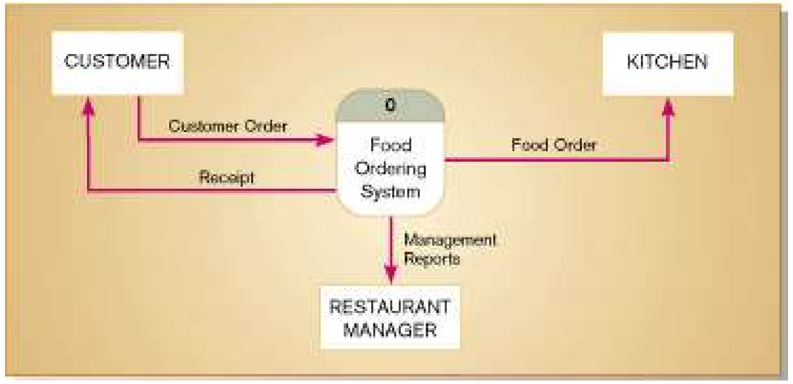
- It is really your decision
- But at some point, the process will be so trivial that it need not be decomposed any further
- Such a process is called a **primitive process**
- **Lowest level is called a primitive DFD**

Rules of decomposition

- When decomposing you the inputs and outputs should be conserved.
- I.e. A decomposed process must have the same inputs and outputs as the pre-decomposed process
- This is called Balancing a DFD

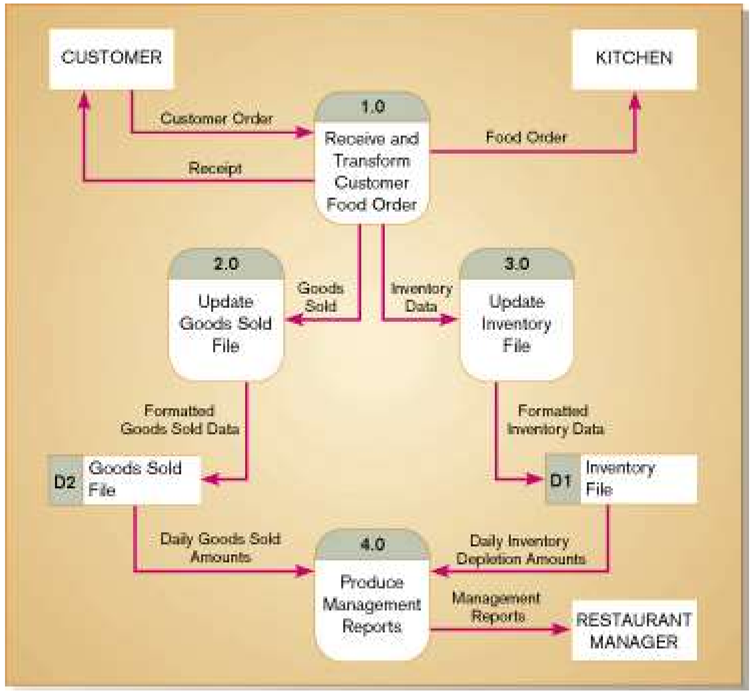
Balancing DFDs

- When decomposing a DFD, you must conserve inputs to and outputs from a process at the next level of decomposition
- This is called balancing
- Example: Hoosier Burgers
  - In Figure, notice that there is one input to the system, the customer order
  - Three outputs:
    - Customer receipt
    - Food order
    - Management reports



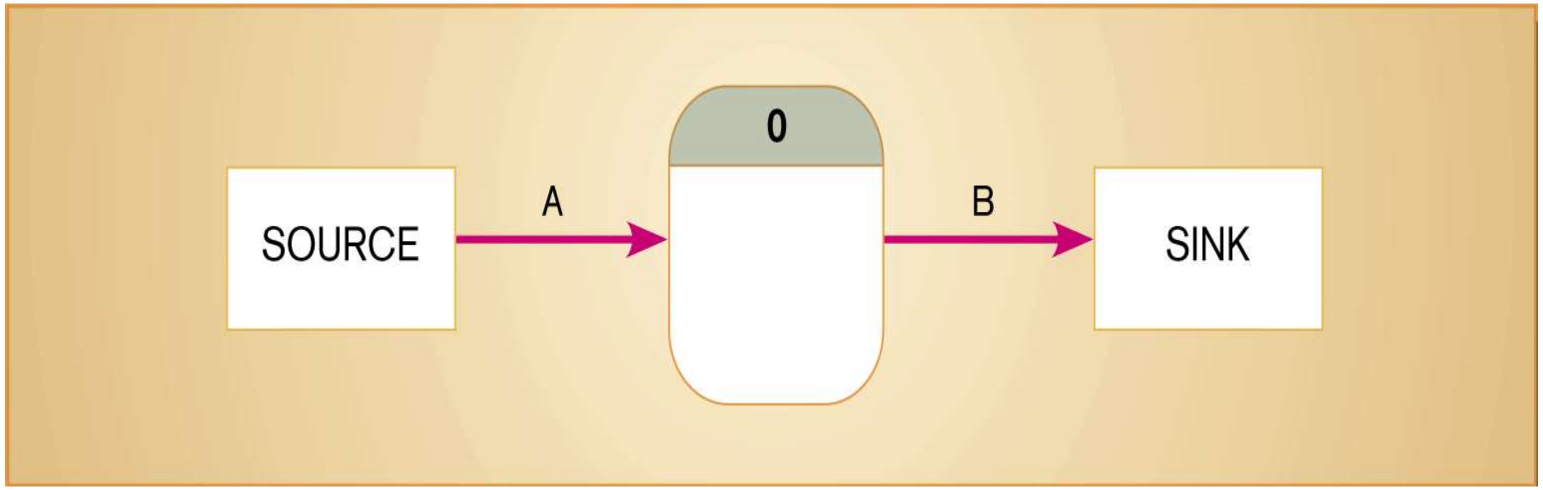
Balancing DFDs

- Example (Continued)
  - Notice the figure. We have the same inputs and outputs
  - No new inputs or outputs have been introduced
  - We can say that the context diagram and level-0 DFD are balanced



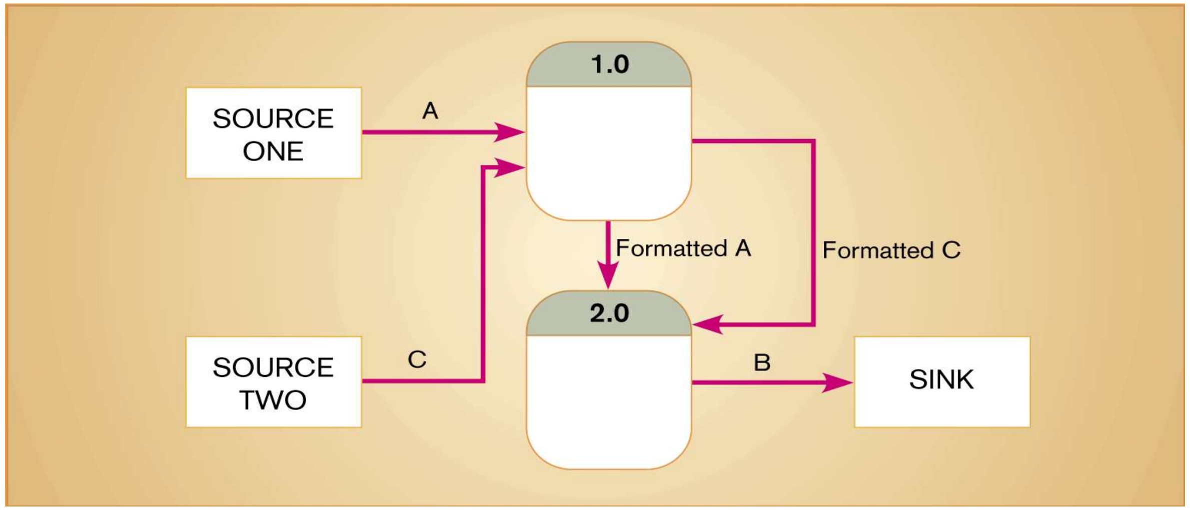
Balancing DFDs

- An unbalanced example
  - In context diagram, we have one input to the system, A and one output, B



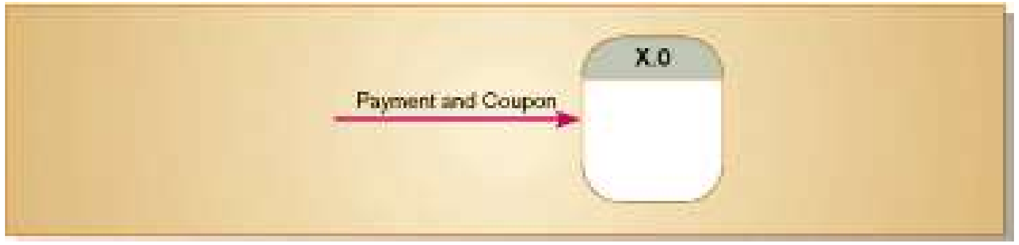
Balancing DFDs

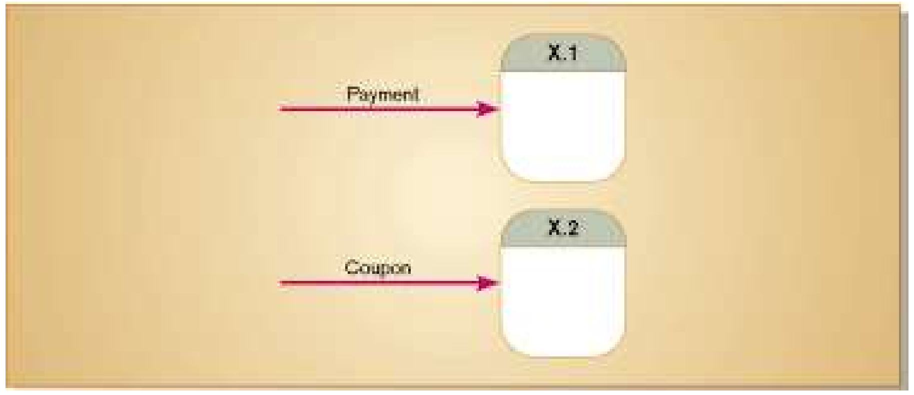
- Level 0 diagram has one additional data flow, C
- These DFDs are not balanced



Balancing DFDs

- We can split a data flow into separate data flows on a lower-level diagram





Rules for Data Flows

- Data flow has only one direction of flow between symbols (unidirectional)
- Data is moved from Source (noun) and to a Sink (noun) by a Process
- Data cannot go directly back into same process it leaves

Rules for Data Flows

- Data in Data Store is moved by a Process
- Data cannot move directly from a source to a data store
- Data cannot move directly from a data store to sink
- Data cannot move directly from one data store to another

Rules for Processes

- Each process must have a unique name
- Processes must have at least one input and one output data flow
- At the lowest level DFD, every process should perform only one well-defined function
  - usually indicated by a single input and a single output
- Primitive Processes
  - processes on the lowest level of the DFD

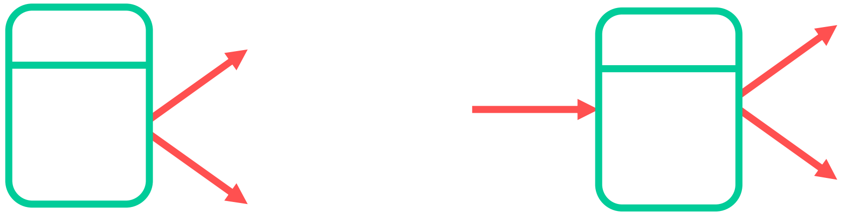
Rules of Sources/Sinks

- Data cannot move directly from a source to sink

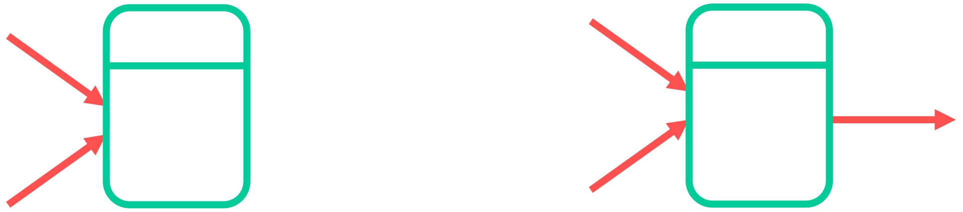
DFD diagram için:
1) Her process için en az 1 input gelmeli ve 1 output çkmal
2) Data storelar dorudan birbirine data gonderemez process olmal arada.
3) Gelen data split edilebilir farkl processler için.

Incorrect and correct DFDs

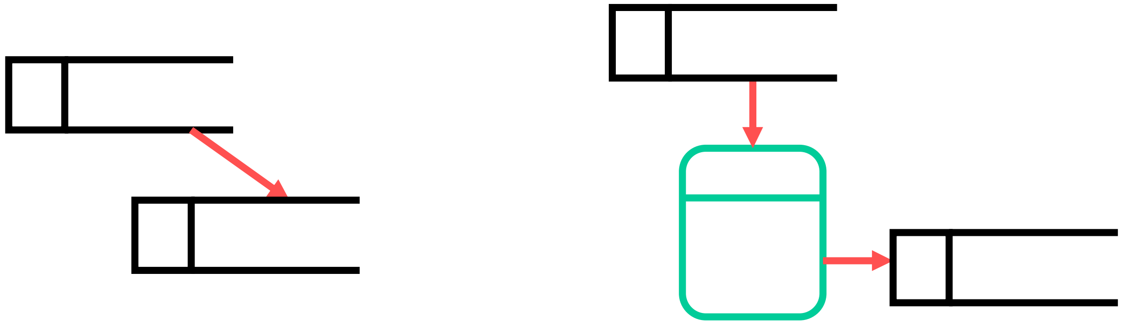
Miracle process not allowed



Black hole process not allowed

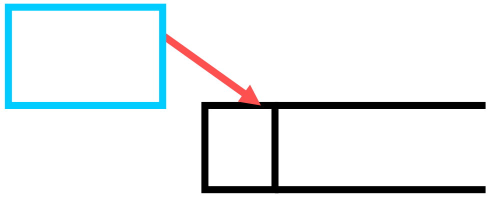


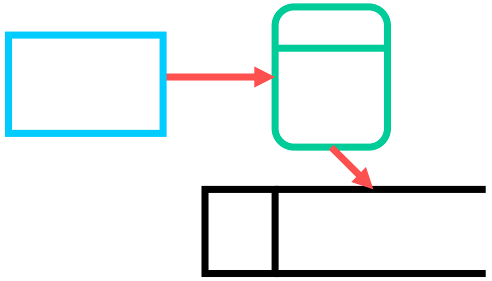
Data must be moved from one data store to another by a process



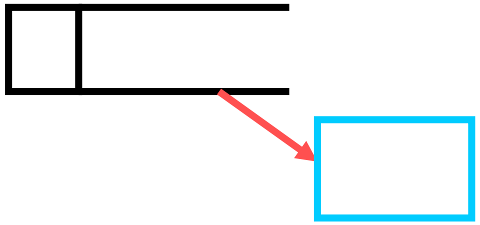
Incorrect and correct DFDs

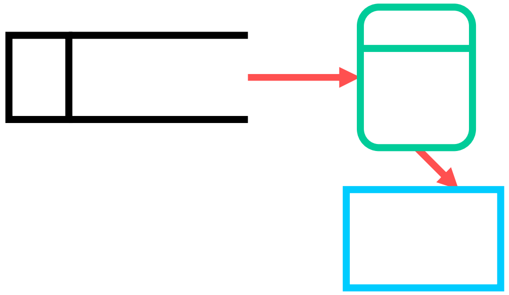
Data must be moved from Source to a data store by a process



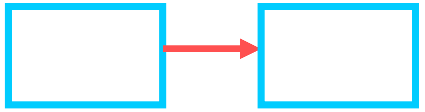


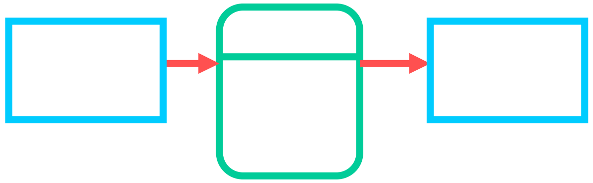
Data must be moved from a data store to a sink by a process





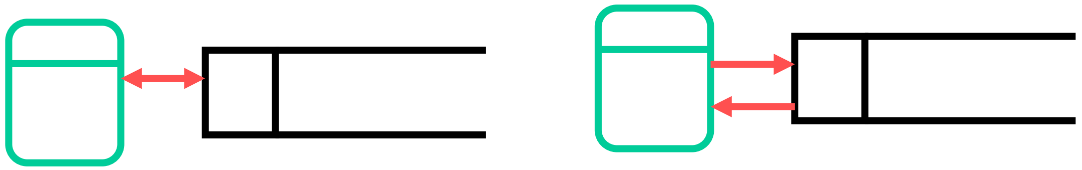
Data must be moved from Source to Sink by a process





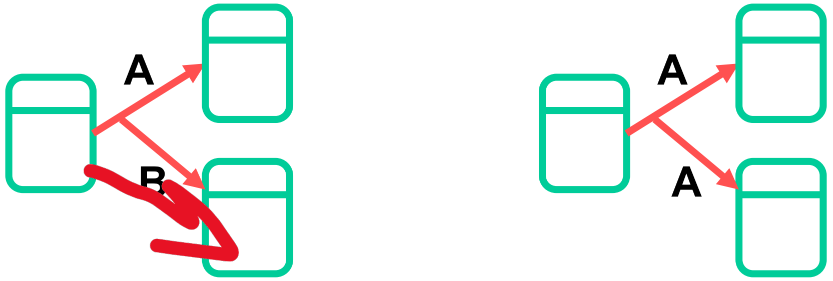
Incorrect and correct DFDs

Data flow is unidirectional

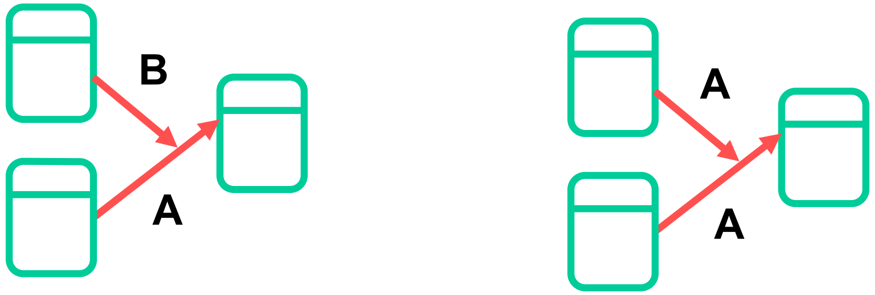


Same processe dönemez yazyor yukarda sor??

Fork not allowed if outgoing dataflow are different

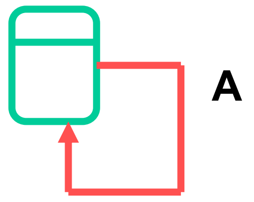


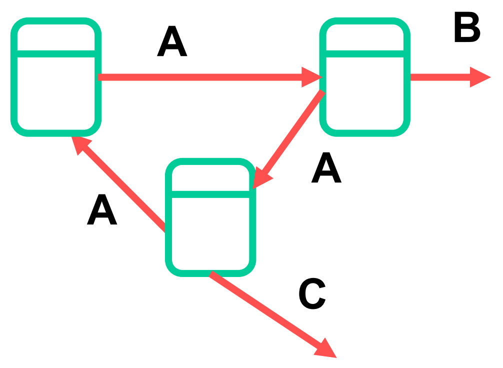
Join allowed only for the same data.



Incorrect and correct DFDs

Data cannot flow directly back to the same process.





Four Different types of DFDs

- Current Physical
  - Process label includes an identification of the technology (people or systems) used to process the data
  - Data flows and data stores are labeled with the actual name of the physical media on which data flow or in which data are stored
- Current Logical
  - Attempt to show the essence of the system without regard to the actual physical implementation. Physical aspects of system are removed as much as possible
  - Current system is reduced to data and processes that transform them. In a Logical DFD, Data Stores, Data Flows and Processes are independent of media.

Four Different types of DFDs

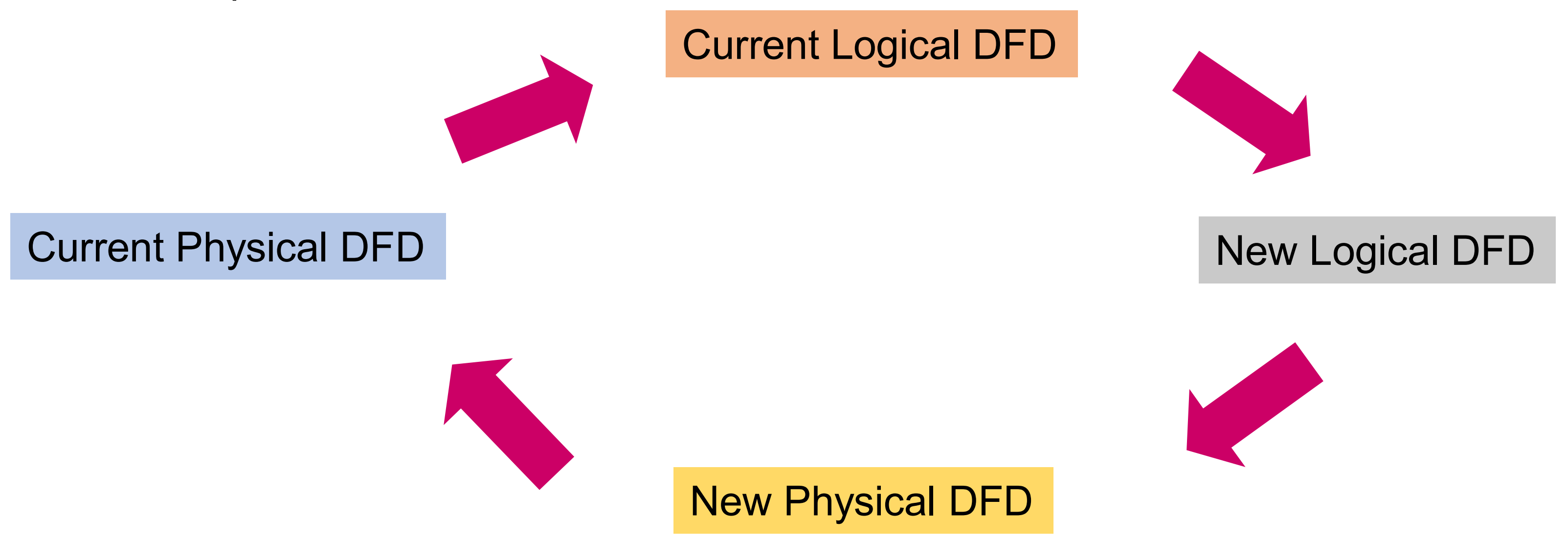
- New Logical
  - Includes additional functions
  - Obsolete functions are removed
  - Inefficient data flows are reorganized
- New Physical
  - Represents the physical implementation of the new system

How are DFDs used?

DFDs are used as an aid for understanding the existing system

They are then used to analysis to look for improvements

Here is the sequence



Guidelines for Drawing DFDs

1. Completeness
   - DFD must include all components necessary for system
   - Each component must be fully described in the project dictionary or CASE repository
2. Consistency
   - The extent to which information contained on one level of a set of nested DFDs is also included on other levels

Guidelines for Drawing DFDs

3. Timing
   - Time is not represented well on DFDs
   - Best to draw DFDs as if the system has never started and will never stop.
4. Iterative Development
   - Analyst should expect to redraw diagram several times before reaching the closest approximation to the system being modeled

Guidelines for Drawing DFDs

- Rules for stopping decomposition
  - When each process has been reduced to a single decision, calculation or database operation
  - When each data store represents data about a single entity
  - When the system user does not care to see any more detail

Guidelines for Drawing DFDs

- Rules for stopping decomposition (continued)
  - When every data flow does not need to be split further to show that data are handled in various ways
  - When you believe that you have shown each business form or transaction, on-line display and report as a single data flow
  - When you believe that there is a separate process for each choice on all lowest-level menu options

Using DFDs as Analysis Tools

- Examine DFDs for:
  - redundant data flows
  - data that are captured but not used
  - data that are updated identically in more than one location
  - excessive processing steps
- Compare new with old logical DFD
  - possible component reuse
- Gap Analysis
  - The process of discovering discrepancies between two or more sets of data flow diagrams or discrepancies within a single DFD
- Inefficiencies in a system can often be identified through DFDs